Measuring and Modelling Delays in Robot Manipulators for Temporally Precise Control using Machine Learning.

Andersen, Thomas Timm; Amor, Heni Ben; Andersen, Nils Axel; Ravn, Ole

Published in:
Proceedings of IEEE ICMLA'15

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

Measuring and Modelling Delays in Robot Manipulators for Temporally Precise Control using Machine Learning

Thomas Timm Andersen*, Heni Ben Amor†, Nils Axel Andersen* and Ole Ravn*
*Department of Automation and Control, DTU Electrical Engineering
Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark. {ttan, naa, or}@elektro.dtu.dk
†Institute for Robotics and Intelligent Machines, College of Computing
Georgia Tech, Atlanta, GA 30332, USA. hbenamor@cc.gatech.edu

***Abstract*—Latencies and delays play an important role in temporally precise robot control. During dynamic tasks in particular, a robot has to account for inherent delays to reach manipulated objects in time. The different types of occurring delays are typically convoluted and thereby hard to measure and separate. In this paper, we present a data-driven methodology for separating and modelling inherent delays during robot control. We show how both actuation and response delays can be modelled using modern machine learning methods. The resulting models can be used to predict the delays as well as the uncertainty of the prediction. Experiments on two widely used robot platforms show significant actuation and response delays in standard control loops. Predictive models can, therefore, be used to reason about expected delays and improve temporal accuracy during control. The approach can easily be used on different robot platforms.**

***Keywords*-Robot control; Automation; Machine learning algorithms;**

I. Introduction

For robots to engage in complex physical interactions with their environment, efficient and precise action generation and execution methods are needed. Manipulation of small objects such as screws and bolts, for example, requires spatially precise movements. However, in dynamically changing environments, spatial precision alone is often insufficient to achieve the goals of the task. In order to intercept a rolling ball on the table, for instance, a robot has to perform *temporally precise* control—the right action command has to be executed at the right time. Yet, by their very nature, actuation commands are never instantaneously executed.

Delays and latencies, therefore, play an important role in temporally precise control and can occur at different locations in the robot control loop. *Actuation delay* is the delay type that most roboticists are aware of. When an action command is sent to the robot's controller, it takes a short while to process the command and calculate the required joint motor input. Imagine a welding robot with an uncompensated actuation delay of 50 ms, working an object on a fairly slow-moving conveyor belt with a speed of 0.5 m/s. The incurred delay would result in a tracking error of 2.5 cm, which could easily destroy a product, or at the very least result in a suboptimal result.

Figure 1. Temporally precise control of an industrial robot is realized by modelling the inherent delay in the system. The picture depicts a fast robot movement during data acquisition. Recorded data is processed using machine learning algorithms to generate predictive models for system and response delay.

A different type of delay is the *response delay* which measures the amount of time until a real-world event is sensed, processed and updated in memory. Response delay is usually assumed zero, as one would naturally assume that this is sampled and transmitted instantaneously whenever a motion occurs. However, since there is a sampling clock and since the controller also needs some time to pack the data for transmission, the response delay can be a non-negligible amount of time. An important implication of the response delay is the discrepancy between the robot's belief of its own state and the true value of that state. When data is received from the robot, indicating that the robot is at a certain position moving with some velocity, the data is in reality describing a state in the past.

In order to effectively act in dynamic environments and reason about timing, a robot has to be aware of both the actuation delay as well as the response delay. Sadly, such information is not readily accessible form the robotics company, and no method is currently available for identifying it. This has lead many researches to develop their own

controllers, but this is rarely an opportunity for industrial users.

Safety during operation is the most crucial issue for robot controllers, but each robotic company may has different strategies which affect the architecture of the robot controller. It is therefore necessary to consider the controller as a black box from which we must learn the controller-dependent delay characteristics. Direct measurement of these delays is typically difficult, since the different delay types are convoluted and hard to separate. An important challenge is therefore the question of how to separate these two delays as, depending on the executed task, a robot has to compensate for a different type of delay.

In this paper, we present a methodology for measuring and modelling the inherent delays during robot control. We introduce an experimental setup which allows us to collect evidence for both the actuation delay, as well as the response delay. The collected data is then used to learn controller-dependent predictive models of each type of delay. The learned predictive models can be used by a robot to reason about timing and perform temporally precise control.

The contributions of this publication are three-fold. First, we provide a generic method for measuring the actuation and response delay of a robot manipulator. Due to its data-driven nature, the method can be used on a variety of actuators. Second, we show how existing machine learning methods can be used to model and predict the inherent delay. Finally, we show modelling results for two widely used robot platforms, namely the Kuka KR 5 Sixx and the Universal Robots' UR10 robot. The acquired data is made publicly available to the robotics community [1].

II. Related work

Modelling time delays is a vital research topic in computer network engineering. In order to ensure fast communication over large computer networks, various models have been put forward to model the mean delay experienced by a message while it is moving through a network [15]. These analytic models typically require the introduction of assumptions, e.g. Kleinrock's independence assumption [11], to make them tractable. Yet, since the network communication is based on a limited number of communication protocols, it is reasonable to use and constantly refine such analytic approaches. Another domain in which latencies and delays play a vital role is *virtual reality* (VR). As noted in [6], latencies lead to a sensory mismatch between ocular and vestibular information, can reduce the subjective sense of presence, and most importantly, can change the pattern of behavior such that users make more errors during speeded reaching, grasping, or object tracking. In VR applications, measuring and modelling delays can be very challenging, since the delay can heavily vary based on the involved software components, e.g., rendering engine, as well as highly heterogeneous hardware components, e.g., data gloves, wands, tracking devices etc. In [6] a methodology for estimating delays is presented, which focuses on VR application domains.

In robotics, the delay inherent to control loops can have a detrimental impact on system performance. This is particularly true for sensor-based control used in autonomous robots. Visual servoing of a robot, for example, can be sensitive to the delays introduced through image acquisition and processing [9]. Similarly, delays in proprioception can produce instabilities during dynamic motion generation. In [2], a dynamically smooth controller has been proposed that can deal with delay in proprioceptive readings. However, the approach assumes constant and known time-delay. A major milestone in robot control with time-delay was the ROTEX experiment [8]. Here, extended Kalman filters and graphical representation were used to estimate the state of objects in space, thereby enabling sensor-based long-range teleoperation. How to effectively deal with such communication delays has been a central research question in robotic tele-operation. Delays in robot control loops are not limited to sensor measurements only. A prominent approach for dealing with actuation delays is the Smith Predictor [17]. The Smith Predictor assumes a model of the plant, e.g. robot system, and can become unstable in the presence of model inaccuracies. A different approach has been proposed in [3]. A neural network was first trained to predict the state of mobile robots based on positions, orientations, and the previously issued action commands. The decision making process was, then, based on predicted states instead of perceived states, e.g. sensor readings. The approach presented in our paper follows a similar line of thought. However, instead of predicting specific states of the robot, we are interested in predicting the delay occurring at different parts of the control loop.

III. Methodology

In this section, we describe a data-driven methodology for modelling delays in robotic manipulators. We show how to acquire evidence for different types of delays and how this information can be used in conjunction with machine learning methods to produce predictive models for control.

A. Measuring the delay

The purpose of the presented method is to establish the actuation and response delay that a high-level control program can expect when issuing commands to a robotic controller. To measure these delays, we need to synchronize the issuing of commands with the control loop of the robot controller. To this end, we use the published current state of the robot, which most controllers send out in each control cycle.

The overall system setup which will be used in the remainder of the paper is depicted in Figure 2 (left). A high-level control program is running on a computer, which sends the commands to the robot control box. The control box,

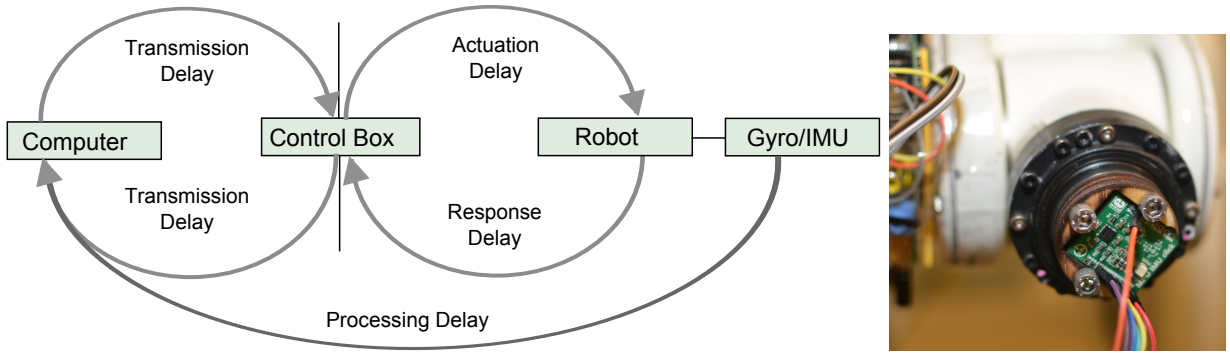

Figure 2. Left: Delays during the control of a robot manipulator. Transmission delay affects information flow between main control computer and the robot control box. Actuation delay and response delay are introduced in the communication between the control box and the physical robot. Right: For delay modelling an external sensor is mounted, e.g. a gyroscope, to measure discrepancies between command times and execution times.

in turn, calculates and issues the low-level commands that drive the robot. The delay between the high-level controller and the control box will be referred to as the *transmission delay*. The transmission delay has already been extensively studied in computer networking [15] and will thus not be treated in this paper. It is particularly crucial in tele-operation scenarios, in which the high-level controller and the robot control box may be separated by thousands of kilometers.

In this paper, however, we focus on the delays incurred between the control box and the robot manipulator. A command that is received by the control box from the high-level program at time $t = 0$ is typically only executed after a delay of ϵ_1. This is the actuation delay. Similarly, once a command is executed by the robot at time $t = \epsilon_1$, it takes another delay of ϵ_2 until the motion is reflected in the controllers memory and transmitted to the high-level program running on the central computer. This is the response delay.

The fundamental idea of our approach is to compare time stamps at the moment a command is issued, the moment the command is executed, and the moment the command gets reflected in the published state of a robot. To this end, it is important to know the ground truth about the true timing of the robot movement. This is realized using an external apparatus in our setup, e.g., a gyroscope or accelerometer, see Figure 2 (right).

1) Determining ground truth: Since we want to measure the delay of the robot, we need a reliable and accurate method of measuring robot motion. The method needs to measure the current motion without adding a significant delay of its own. This can be achieved by imposing a significantly higher sampling rate than the robot controller.

We use microelectromechanical (MEMS) gyroscopes, or angular rate sensors, for the revolving joints, and MEMS accelerometers for prismatic joints. They offer very high sampling rates of several orders of magnitude higher than many robot controllers publish (e.g. several kHz for affordable sensors), and practically no delay from motion to available measurement. Such sensors cannot readily be used to infer where in the kinematical chain a motion has occurred, hence measurements have to be performed a single joint at a time. Gyroscope measurements often come with significant noise, while accelerometer measurements suffer from drift. However, both of these issues can be compensated for using simple offline filtering in-between measurement and training the model.

2) Acquiring measurements: As mentioned before, our approach is based on comparing time stamps throughout the robot control loop. To this end, we use the published state from the robot as the main sample clock and reference. An experimental trial starts at $t = 0$ upon reception of a first package from the controller. The system time stamp is recorded as soon as data is read, and the byte-encoded package is stored for later parsing to extract the current joint state. Upon reception, a command is sent instructing the robot to start moving a single joint, which we monitor with our angular rate sensor or accelerometer. The commanded movement consists of a rapid acceleration in one direction, followed by a fast deceleration before returning to return to the starting pose. The entire motion trial takes about a second, and all packages received until the robot stands still are stored. Sensor readings from the external sensor are stored by the central computer in order to identify the ground truth time stamp of the moment in which the robot moved.

There are several perturbations that can lead to variations in the incurred delays, in particular physical perturbations. For instance, the force resulting from the gravitational pull on the robot varies with the joint configuration of the robot, just as the direction of motion effects whether the motor needs to work against or along gravity. The different size of motors and gearing in the robot also yields varying results. These perturbations lead to varying static and kinetic friction in the moving parts of a robot. This variation in turn leads to a varying actuation delay.

As the magnitude of the static friction is usually larger than that of the kinetic friction, we assume that the delay is

mostly affected by the robot's joint configuration when the motion starts. We assume that the effect by the other joints during a motion after the static friction has been overcome can be neglected. A similar assumption of joint independence if often employed on the joint position controller when using Independent joint control [14].

To acquire a representative data set for modelling delays, we therefore need to map out the delay of each joint for all the different joint combinations, moving in both positive and negative direction. To capture variance in the delay, each combination of joint configuration and direction should be measured several times.

3) Filtering data and computing delays: When extracting the delays, we evaluate the difference between the recorded data. Before doing that, thought, we use a high order low-pass FIR filter (Figure 3) on the data from the angular rate sensor and correct for any drifting of the accelerometer, based on data recorded while the sensor was held stationary on the robot.

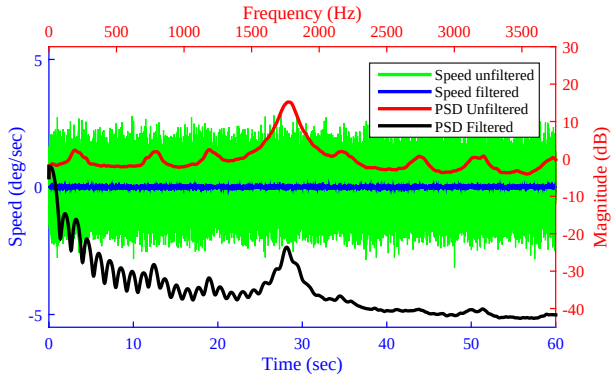

Figure 3. Gyroscope readings are filtered using a FIR filter. A 60 second datastream (green), recorded without moving the robot, is passed through the filter to remove noise (blue). The frequency component of the data before and after filtering is shown in red and black, calculated using Welch's Power Spectral Density (PSD) estimate [18]

To calculate the delay, we evaluate our two data series generated in each trial; the speed output from the robot controller and the filtered sensor data. The actuation delay is the difference between the moment a command is sent to the robot and the moment a sensor registers the motion, while the response delay is the difference between the moment a sensor registers motion and the moment it is reflected in the robot's current speed data. Both are calculated while taking into account the transmission delay from Figure 2 (left). Even when filtering out the noise, it can be challenging to establish the exact moment in time when the sensor determines that a motion has started as the measured speed is hardly ever zero. Instead we identify extrema of our recorded data to detect the time difference between the set target speed, the measured speed, and the reported current speed.

B. Learning Predictive Models of Delay

Next, we want to use the recorded data in order to learn predictive models of robot latencies. Once a predictive model is learned, it can be used by a robot to infer the most likely delay in a given situation. A common approach in robot control is to use a path planner running on the central computer to generate a starting joint configuration and an execution time of the trajectory. To find the actual real time that the robot will use to get to the goal state, we can query the learned predictive models for each moving joint. The individual delay is then added to the execution time of each joint to identify the real execution time.

As input features for the model we use the starting joint configuration of the robot. As mentioned before, forces acting on the robot vary depending on the joint configuration and impact in particular the actuation delay. The output of the model is the expected delay. We learn individual models for the actuation delay and the response delay, since these two delays are unrelated. In line with the assumption of independence between the joints, a separate model is learned for each joint. Introducing the above structured approach, allows for accurate predictions of the delay. To evaluate how a unified model, predicting the delay of all joints, performs, such a model is also trained.

The goal of learning is to generate predictive models that can generalize to new situations and lead to accurate predictions of the expected latencies. To this end, we use three different machine learning methods, namely neural networks (NN) [4], regression trees (RT) [5], and Gaussian processes (GPR) [12]. We use these methods as they can all effectively recover nonlinear relationships between input and output data.

In our specific implementation, we used a feed-forward neural network with 30 neurons in a single hidden layer. Learning was performed using the Levenberg-Marquardt [10] algorithm. In contrast, the regression tree method hierarchically partitions the training data into a set of partitions each of which is modelled through a simple linear model. Both NNs and the RTs produce a single result and do not provide information about the uncertainty in the predicted value. In contrast to that, GPR can learn probabilistic, non-linear mappings between two data sets. Due to the inherent noise and related phenomena, uncertainty handling is a crucial issue when dealing with delays.

By providing the mean and the variance of any prediction, the GPR approach allows us to reason about uncertainty of our prediction. Together, mean and variance form Gaussian probability distribution indicating the expected range of predictions. This information can potentially be exploited to generate upper- and lower-bounds for the expected delays, which is in contrast to both NN and RT.

As both NN, RT, and GPR are well known machine learning methods and we do not add anything to these

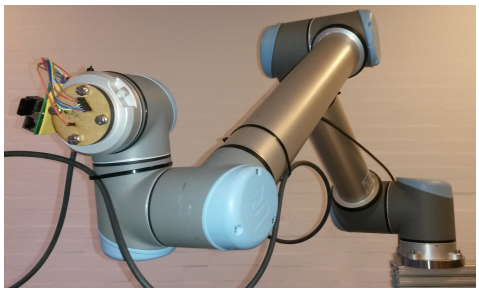

Figure 4. The Universal Robot UR10 with mounted measuring equipment. The enclosure keeps the sensor at a stable temperature thus avoiding temperature-related drift in measurements.

methods, the theory behind them will not be covered further in this paper.

IV. Results

A. Experimental setup

In our experiments, we model the performance of both a Kuka KR 5 sixx (Figure 1) and a Universal Robot UR10 (Figure 4). To generate the training data, we mounted a MPU6000 combined angular rate sensor and accelerometer to the end-effector. To avoid temperature-related drifts, we

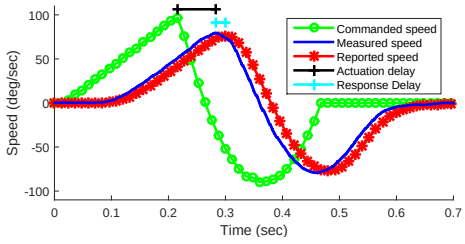

Figure 5. Typical plot of logged data from a single trial. 33,500 trials were completed on each robot.

mount the sensor on the robot in an enclosure with low heat conductivity and then let the sensor warm up before measurements. Since these robots only have revolute joints, only the angular rate sensor is used, which outputs data at a rate of 8 kHz.

To collect the data used for training the model, we perform a series of short trials, wherein the robot is commanded to perform a fast acceleration and deceleration motion. For controlling the Kuka robot, the Kuka RSI [7] protocol is used. It operates with a sample rate of 83.3 Hz (12 ms). As argued in [13], the UR10 is controlled using URScript SpeedJ commands. It operates with a sample rate of 125 Hz (8 ms).

An example trajectory, along with typical outcome of a trial, can be seen on Figure 5. The plots clearly show a significant time difference in the commanded speed, the reported speed and the measured speed. To capture the variations of the delays, we performed trials on 4 different joints, moving 10 times in both positive and negative direction in 1,920 different joint configurations. A total of 33,500 trials were performed on each robot in order to generate a comprehensive dataset, to be released to the public [1]. For purposes of machine learning, only a subset of the data was later used.

To be able to compare the performance of the two robots, we used the same 1,920 physical joint configuration (i.e. all links vertical) for both robots rather than using the same joint values. This is a necessity since the Denavit-Hartenberg parameters of the robots are not identical and the home position varies, thus positive joint rotation on one robot might lead to negative joint rotation on the other. Sampling only a subspace of the robots' total workspace does not introduce bias in the data, but rather limits the model to predict delays within that subspace. By sampling more poses, the model can routinely be extended to cover the entire workspace if needed.

B. Delay output

As explained in Section III-A3, delays are determined by evaluation of the extrema of the recorded motions. An example of how the delays vary for the two robots can be seen on Figure 6. The distribution of the actuation delays can be seen on Figure 7, while a boxplot showing the individual delays per joint is shown on Figure 8. The same plots for the reaction delays can be seen on Figure 9 and Figure 10, respectively.

C. Model comparison

The extracted delays were used to train and validate models based on different machine learning algorithms, namely NN, RT, and GPR. For the NN and RT algorithms, we used the standard MatLab implementation, while we used GPstuff [16] for the GPR implementation. The starting joint configuration, the actuated joint, and the rotational direction were used as input. The delays that were measured at each input combination were used for training and testing, using k-fold cross validation with 10 folds to limit overfitting the data and to give an insight on how the model will generalize to an independent dataset. The mean squared error (MSE) from each fold were averaged together and used as a measure of how well the model predicts delays. Models for predicting both the delay of individual joints, as well as a combined model that can predict the delay of all joints were trained. The mean error of each model is derived by taking the square root of the MSE and is shown in Table I and II. The tables also shows the resulting mean error if the delay was assumed that of the median of the corresponding boxplots. This gives an indication of the performance of the trained models. Lower values indicate better generalization capabilities, while larger mean error values indicate poor prediction performance.

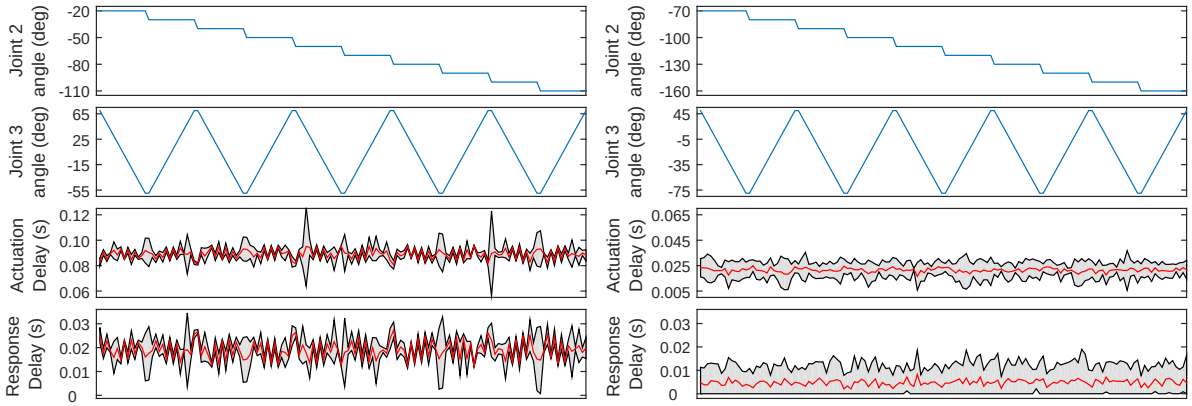

Figure 6. Actuation and response delay for joint 3 moving in positive direction as a function of varying joint 2 and 3. The red graph is the mean and the gray area is ± 2 standard deviations, corresponding to a 95% confidence interval. Note the different y axis interval. Left: Kuka. Right: Universal Robot.

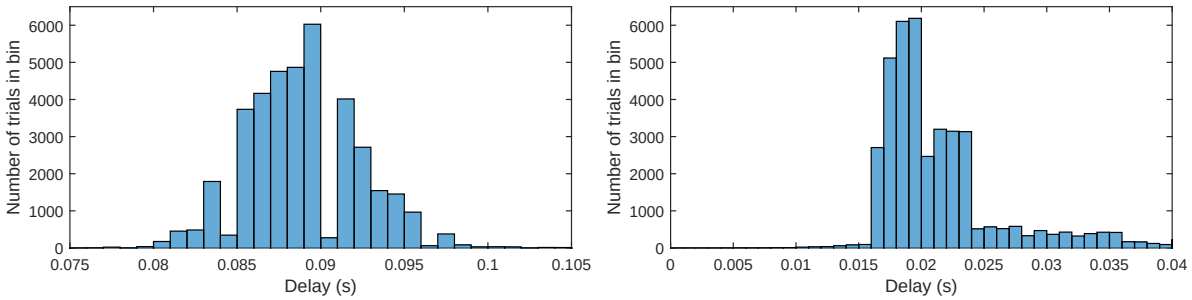

Figure 7. Combined distribution of the actuation delay of all joints. Note the different x axis interval. Left: Kuka. Right: Universal Robot.

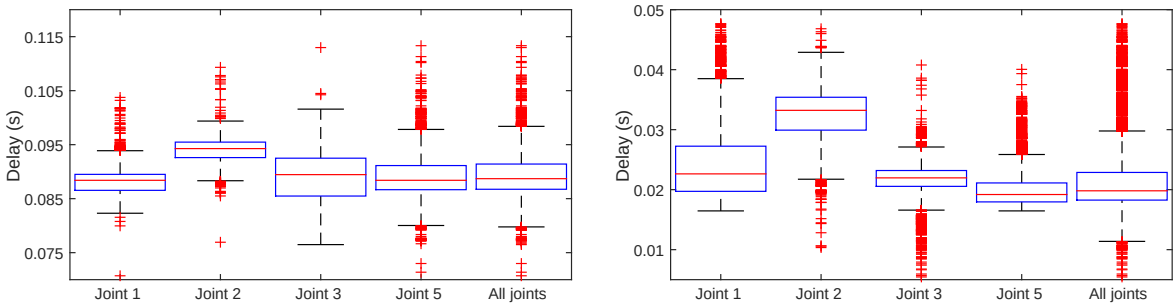

Figure 8. Boxplot of individual joint's actuation delay. Note the different y axis interval. Left: Kuka. Right: Universal Robot.

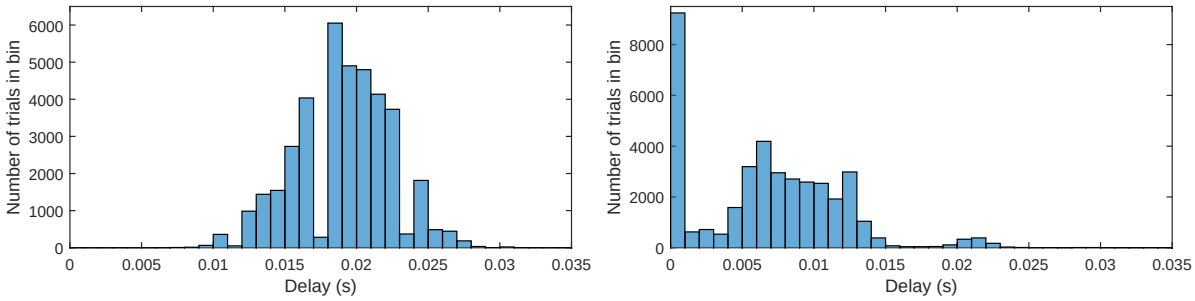

Figure 9. Combined distribution of the response delay of all joints. Left: Kuka. Right: Universal Robot.

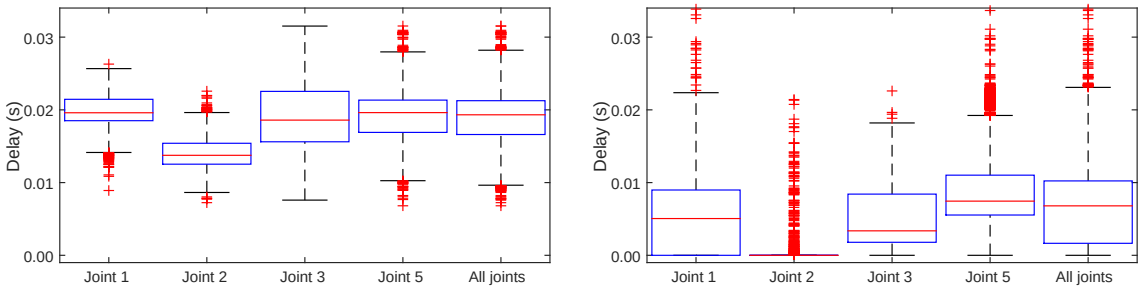

Figure 10. Boxplot of individual joint's response delay. Left: Kuka. Right: Universal Robot.

Table I
MEAN ERROR IN MILLISECONDS OF MODEL FIT FOR ACTUATION DELAY.

Kuka	Joint 1	Joint 2	Joint 3	Joint 5	Combined	Average
Median	2.27	2.68	4.61	3.51	3.67	3.35
NN	1.79	2.55	4.74	3.37	3.21	3.13
RT	1.99	2.48	3.74	3.76	3.33	3.06
GPR	1.85	2.27	4.30	3.39	3.48	3.06
UR	Joint 1	Joint 2	Joint 3	Joint 5	Combined	Average
Median	6.18	4.64	6.08	2.59	5.33	4.96
NN	4.08	5.32	3.86	2.49	3.12	3.77
RT	4.63	5.41	4.28	2.72	3.42	4.09
GPR	5.01	4.89	3.68	2.47	3.36	3.88

Table II
MEAN ERROR IN MILLISECONDS OF MODEL FIT FOR REACTION DELAY.

Kuka	Joint 1	Joint 2	Joint 3	Joint 5	Combined	Average
Median	2.13	2.37	4.30	3.09	3.33	3.05
NN	1.70	2.32	4.68	2.22	2.36	2.65
RT	1.86	2.21	3.62	2.31	2.37	2.47
GPR	1.63	2.17	4.23	3.11	2.63	2.75
UR	Joint 1	Joint 2	Joint 3	Joint 4	Combined	Average
Median	4.82	2.00	4.08	4.90	5.09	4.18
NN	4.11	2.48	4.24	5.22	4.84	4.01
RT	4.66	2.44	4.47	5.84	5.33	4.35
GPR	3.88	2.44	3.75	5.20	5.05	3.82

V. DISCUSSION

A. Evaluating the two robots' delays

As it can be seen on Figure 7, the actuation delay of the Kuka is significantly higher than on the Universal Robot, even factoring in the higher sample period; the average delay for the Kuka is 7.5 sample periods vs. 2.5 sample periods for the UR. If we relate the figure to the example from the introduction, where a welding robot need to weld an object on a conveyor belt moving at 0.5 m/s, our claim that it is important to compensate for the delay is clearly justified. The Kuka robot would, without compensation, make a welding seam displaced $4.5cm \pm 0.5cm$ from the target, while the Universal Robot would miss with $0.75cm - 1.25cm$.

A deeper look into the actuation delays, which is on Figure 8, shows that the delays in general only vary with a few ms for each joint. Using our method for measuring the delay and assuming the delay constant at the median of each boxplot would thus decrease the error to within 0.3 cm for a delay within ± 6 ms. If we include the whiskers of the boxplot, corresponding to $\sim \pm 2.7\sigma$ or 99.3% of the data, the worst case error would within 0.85 cm for a delay within ± 17 ms.

Figure 8 also shows that on both robots, it is joint 2 that has the highest delay. This is the shoulder joint, and the one that lifts the most. This supports our theory that gravity influences the actuation delay. Figure 10 suggests that the response delay on the other hand is not varying between the joints. This is not surprising, as the response delay, as mentioned previously, is largely incurred by the sampling clock, packing of data and transmission. This most likely happens simultaneously for each joint.

The seemingly correlation between actuation and response delay on Figure 6 is a consequence of the relatively low temporal resolution of the robot controller data. This is also why it is more dominant on the Kuka robot. As the sum of the actuation and response delay will always be a multiple of the sample period, an actuation delay a few ms below the mean at a specific pose will result in a response delay a few ms above the mean at that pose.

A surprising finding on Figure 9 is that the response delay for the Kuka robot is more than one sample period, which suggests that sampling and transmission of data takes place in separate sample clock cycles.

B. Evaluating the models' performance

All of the models are able to predict the delays very accurately to within a mean error of $5ms$ and it is thus difficult to say anything conclusive about which model is best. Though all of the models would have a mean error less than 0.35 cm if used for a typical task like welding, which is an improvement of more than a factor 12 for the Kuka robot and almost a factor 3 for the Universal Robot, compared to using the controllers and not assuming any delay. Comparing the learned models with measuring the delay and assuming it to be static shows an improvement

between 6 and 24%.

The response delay for the Universal Robot shows the least benefit from modeling. This is most likely due to the fact that the spread of the delays are so small. The missing improvement with machine learning is thus a result of the median delay yield a very good guess, and not a result of the models being poor at learning those delays.

It is worth noticing that the mean error in some cases are significantly higher for the Universal Robot models than those of the Kuka robot. This correspond with Figure 6, where the confidence interval is much broader for the Universal Robot than for the Kuka.

It should also be noted that GPR does not only supply a prediction of the delay, but also outputs a measure of uncertainty, which is not reflected in the tables. For the Universal Robot's large variance, this is certainly an added bonus.

VI. Conclusion

In this paper we presented a methodology for measuring and separating actuation and response delays in robot control loops. In addition, we introduced a data-driven approach for modelling inherent delays using machine learning algorithms. We showed that the introduced models can be efficiently used to predict occurring delays during temporally precise control.

Real world experiments were used to identify latencies in two widely used robot platforms. The measured delay showed a large potential for improving temporal precision, with more than a factor 12 improvement for one of the robots.

All the employed machine learning algorithms showed similar abilities to further improve the accuracy, with no algorithm showing significantly better accuracy than the others. Still, Gaussian processes seem to be better suited for this task, since they provide a probability distribution over the expected delay. In turn, such a distribution can be used to reason about upper- and lower-bounds in temporal precision.

In our future work we will investigate how inverse models of time delay can be learned. Given a specific time constraint during a control task, an inverse model can be queried for the most appropriate action which will meet the goals of the task while ensuring time constraints.

References

[1] http://aut.elektro.dtu.dk/staff/ttan/delay.html.

[2] S. Bahrami and M. Namvar. Motion tracking in robotic manipulators in presence of delay in measurements. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3884–3889, May 2010.

[3] S. Behnke, A. Egorova, A. Gloye, R. Rojas, and M. Simon. Predicting away robot control latency. In *RoboCup 2003: Robot Soccer World Cup VII*, Lecture Notes in Computer Science, pages 712–719. Springer Berlin Heidelberg, 2004.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[6] M. Di Luca. New method to measure end-to-end delay of virtual reality. *Presence: Teleoper. Virtual Environ.*, 19(6):569–584, December 2010.

[7] KUKA Robot Group. *KUKA.Ethernet RSI XML 1.1*, kst ethernet rsi xml 1.1 v1 en edition, 12 2007.

[8] G. Hirzinger, K. Landzettel, and Ch. Fagerer. Telerobotics with large time delays-the rotex experience. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 1, pages 571–578 vol.1, Sep 1994.

[9] A.J. Koivo and N. Houshangi. Real-time vision feedback for servoing robotic manipulator with self-tuning controller. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(1):134–142, Jan 1991.

[10] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[11] A. Popescu and D. Constantinescu. On kleinrocks independence assumption. In DemetresD. Kouvatsos, editor, *Network Performance Engineering*, volume 5233 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg, 2011.

[12] C. E. Rasmussen and Ch. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[13] O. Ravn, N. A. Andersen, and T. T. Andersen. Ur10 performance analysis. Technical report, Technical University of Denmark, Department of Electrical Engineering, 2014.

[14] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*. John Wiley & Sons New York, 2006.

[15] A. S. Tanenbaum and D. J. Wetherall. *Computer Networks*. Prentice Hall, 5th edition, 2011.

[16] Jarno Vanhatalo, Jaakko Riihimäki, Jouni Hartikainen, Pasi Jylänki, Ville Tolvanen, and Aki Vehtari. Gpstuff: Bayesian modeling with gaussian processes. *The Journal of Machine Learning Research*, 14(1):1175–1179, 2013.

[17] P.D. Welch. A controller to overcome dead time. *ISA Journal*, 6(2):28–33, 1959.

[18] P.D. Welch. A direct digital method of power spectrum estimation. *IBM Journal of Research and Development*, 5(2):141–156, 1961.